Designing with SMBus 2.0

Dale Stolitzka
Analog Devices, Inc.

January 29, 2001
Acknowledgements

Materials and content for this presentation has been compiled from a variety of SMBus sub-committee members. The author wishes to acknowledge and thank contributors: Steve Williams, Cliff Laney and John Milios for their efforts on the development of the SMBus 2.0 specification.
Agenda

• Problem Statement
• SMBus on PCI and mini-PCI connector
• SMBus Addressing Problem & ARP
• Physical Layer – AC & DC Changes
• Packet Error Checking Review
• Impact on SMBus 1.0 & 1.1
• Summary
Problem Statement

- SMBus allows signaling between PC components
  - Pre-boot or when OS is down
  - When the PCI bus is unavailable
- OEMs demand SMBus signaling on expansion cards
- Pig-tails are bad
- How do we use SMBus on PCI?
SMBus on the PCI and mini-PCI Connector

- Specified in the SMBus 2.0 spec and in an ECR to the PCI 2.2 specification
- SMBus uses 2 pins on the connector
- More robust electrical specifications
- Plug ‘n Play capability to resolve SMBus addresses
  - Address Resolution Protocol (ARP)
SMBus Addressing Problem

- SMBus devices require a bus address
- Old model used fixed addresses
- Expansion cards require dynamically assigned addresses
  - Avoid conflicts with addresses used by motherboard devices
  - Allow for multiple boards of the same type
Address Resolution Protocol (ARP)

• Every SMBus device has a Unique Device ID (UDID)
  • fixed or random
• At start time, the bus is enumerated
  • Some devices may be fixed on motherboard
• Addresses are assigned to all enumerated devices
ARP in the System

• An ARP Agent may run ARP:
  • Software running over a host controller
  • Microcontroller running independently
• The bus may be enumerated at anytime without address assignment
• Software makes address map available to apps
### Unique Device Identifier

<table>
<thead>
<tr>
<th>8 bits</th>
<th>8 bits</th>
<th>16 bits</th>
<th>16 bits</th>
<th>16 bits</th>
<th>16 bits</th>
<th>16 bits</th>
<th>32 bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>Device Capabilities</td>
<td>Version / Revision</td>
<td>Vendor ID</td>
<td>Device ID</td>
<td>Interface</td>
<td>Subsystem Vendor ID</td>
<td>Subsystem Device ID</td>
<td>Vendor-Specific ID</td>
</tr>
</tbody>
</table>

- **Device Capabilities** = specific functions, e.g., supports PEC
- **Version/Revision** = silicon revision, SMBus UDID version
- **Vendor ID** = device manufacturer’s vendor ID
- **Device ID** = device ID assigned by manufacturer
- **Interface** = SMBus version of this device
- **Subsystem Vendor ID** = interface ID for industry group
- **Subsystem Device ID** = device ID for industry group
- **Vendor-specific ID** = 0x00000000 for SMBus 1.x ensures a unique UDID in SMBus 2.0
How UDID is Used in Segment/Device Info

**SMB_INFO Structure**
- SMB_INFO Struc Ver
- SMBus Spec Version
- Segment HW Capability
- Reserved
- Device Count
- Device 0 Info
- Device 1 Info
- ...
- Device n Info

**SMB_Device Structure**
- Slave Address
- Reserved
- Device UDID

**SMB_UDID Structure**
- Device HW Capabilities
  - Version/Revision
  - Vendor ID
  - Device ID
  - Interface
  - Subsystem Vendor ID
  - Subsystem Device ID
  - Reserved

HC supports PEC
Slave device supports PEC
Example of an ARP Agent

- Performs ARP process
  - Build assigned address pool
  - Assigns addresses to SMBus 2.0 devices
- ARP agent may run when...
  - SMBus driver initializes
  - Hot insertion / PnP notification received
  - Resume notification
  - ‘Notify ARP Master’ from SMBus HC (requires HC interrupt support)
Building the Assigned Address Pool

Fixed Slave Addresses
Reserved Address Assignments
ARP Assigned Addresses

Assigned Address Pool

_SBI control method in SMBus CMI 1.0 Spec
Appendix ‘C’ in SMBus 2.0 Spec
Obtained by running ARP
Selecting ARP Slave Addresses

Rules:

• Retain the same slave address if at all possible
• Honor slave address preferences if possible
• For new devices assign unused slave addresses first
• Recycle ‘pre-owned’ slave addresses only as a last resort
Physical Layer Features

• SMBus 2.0 is designed to be backwards compatible
• Bus rate is the same
• Two DC power classes
  • Low-power class ⇒ SMBus v1.1 limits
  • New high-power class ⇒ SMBus v2 (these work on SMBus 1.x systems)
AC Data Rate

- Not really, they refer to different issues
  - **Periodic clock stretching**: $F_{SMB} \propto$ data rate of *typical* data bits
  - **Random clock stretching**: $T_{LOW:SEXT}$ is the total time a slave device may stall the bus
AC Rise/Fall Timing

- DC levels for $t_{\text{RISE}}$, $t_{\text{FALL}}$ are symmetric rather than dependent on $V_{\text{DD}}$

$0.9 \times V_{\text{DD}}$

$V_{\text{IH,MIN}} + 0.15V$

$V_{\text{IL,MAX}} - 0.15V$

$t_{\text{RISE}}$

$t_{\text{FALL,1.x}}$

$t_{\text{FALL,2.0}}$
AC Timeout

• $T_{\text{TImeout}}$ clarified, not changed
• Devices timeout after SMBCLK is low for 25ms
• Host controllers must support $T_{\text{TImeout}}$
DC New High-power Class

- $C_{BUS} \leq 400\text{pF}$
  - $\sim 11\times$ larger than low-power class
- $I_{PULLUP} \geq 4\text{mA}$
DC Back-powering

- Unpowered devices must not load the bus or power-up devices
- $I_{\text{LEAK-PIN}} \leq 10\mu\text{A}$
  - relaxed from SMBus v1.x
Packet Error Checking

- PEC (Packet Error Code) byte appends to message
- Uses CRC-8 polynomial: $C(x) = x^8 + x^2 + x + 1$
- Required for Address Resolution Protocol
# PEC implementation

## Write Byte/Word/Block

- **S**: Slave Address
- **W**: Write
- **A**: Command Code
- **A**: Data Byte
- **P**: Data provided by master transmitter

## Read Byte/Word/Block

- **S**: Slave Address
- **R**: Read
- **A**: Data Byte
- **P**: Data provided by slave receiver

CRC-8 is calculated upon the whole message including Address and R/W bit.
Error Recovery

- Specifies the use of ACK, NAK signaling for flow control and error recovery in bus transactions
- Different than I²C
  - I²C specifies that you can ACK/NAK the following byte
- Works in conjunction with CRC-8
ACK and NAK usage

A NAK by the Slave Receiver during a Write operation before the last byte is interpreted as a transaction ABORT (busy, overflow, unknown command, etc.)

A NAK by the Slave Receiver at the CRC-8 byte means RESEND

The Master Receiver detecting an error in the Packet Check Code, terminates the transaction and repeats the operation
Hardware PEC Generator

- CRC is $C(x) = x^8 + x^2 + x + 1$
- Simple hardware-generated PEC for hardware-based ASIC
PEC with Intelligent Targets

- Intelligent targets (μC) recover errors in other layers
  - Last ACK ⇒ PEC byte received
  - Allows μC core to calculate CRC-8 without stalling the SMBCLK
- More IC selection to OEM

This device is hereby permitted to recover errors in clever ways by virtue of its extraordinary intelligence.
PEC & Intelligent Target Hardware

NAK means PEC is bad in either SMBus 1.1 or SMBus 2.0

ACK means PEC is valid in SMBus 1.1
An ACK by an intelligent SMBus 2 target means the PEC is received but may defer error recovery to a higher protocol layer
Example

*Sensors and Add-in Targets*

- Targets on the motherboard
  - Hardware monitors, EEPROM, etc.
  - Don’t need a programmable address
  - May participate in ARP Discovery
- Targets on add-in cards
  - Must ARP & accept address assignments
Example

Chipset

- Use a STOP bit for message aborts
- Host must watch for slave timeouts
- Use error checking whenever possible
Example

Host/Master clearing the bus

- Host detects timeout
  - Send Stop condition
- Is the bus clear?
  - if SMBDAT is still low
  - Clock the bus until SMBDAT is high
Impact on SMBus 1.0 & 1.1

• SMBus 1.0 & 1.1 are SMBus 2.0 compliant
  • No current part becomes obsolete
  • May be used in current applications
    • In Smart Batteries and motherboard sensors

• SMBus 1.0 and 1.1 parts may not be used on PCI expansion cards
  • ARP and high-power electricals required on PCI and mini-PCI boards
Summary

• SMBus 2.0 is designed to be backwards compatible
• SMBus 2.0 can be used on add-in cards and down on the motherboards
• ARP introduced for v2.0 which requires PEC
• Some changes in AC and DC from SMBus 1.1
• SMBus 2.0 is released and available, now
Collateral

• SMBus website: http://www.smbus.org
• All SMBus specs, 1.0, 1.1, 2.0, SMBus Device Driver External Architecture Specification, SMBus Control Method Interface Specification: http://www.smbus.org/specs/index.html
• PCI SIG website: http://www.pcisig.com
• SMBus ECR to PCI 2.2 Specification
  • check Members area on PCI SIG website
Call to Action

• Device vendors:
  • Implement SMBus 2.0; For legacy devices, plan UDIDs
• BIOS vendors:
  • Implement control methods
• System OEMs:
  • Build SMBus 2.0 into systems now!
  • wire all PCI slots for SMBus 2.0
• Card vendors:
  • Add SMBus 2.0 to cards where it makes sense
Example of an SMBus 2.0 Driver Architecture†